Calculs paramétriques - Distribution de calculs

Résumé:

Certaines études conduisent à effectuer un nombre plus ou moins important d'analyses (plusieurs milliers parfois), correspondant aux variations des paramètres. L'utilisation de calculs standards, consistant à modifier ou créer le fichier de commande pour chacun des jeux de paramètre est fastidieuse, difficile à gérer et source d'erreur. La méthode présentée dans ce document permet de faciliter la mise en œuvre de telles études paramétriques en effectuant un minimum d'interventions.

A partir de l'étude standard et d'un fichier contenant le jeu de paramètres, les calculs sont déclinés et exécutés automatiquement.

Les calculs paramétriques étant indépendants les uns des autres, il est possible d'utiliser les ressources machine disponibles en soumettant en parallèle les calculs.

Table des matières

1 Généralités

1.1 Présentation

Une **étude paramétrique** est une étude standard (onglet `ETUDE`) dans laquelle on souhaite faire varier un ou plusieurs paramètres, comme par exemple :

- Paramètres matériaux : module d'Young, limite élastique,...
- Paramètres géométriques : épaisseur de la coque, section d'une poutre,...
- Paramètres chargements : pression, orientation d'une force, ...
- ...

Si l'on effectue ces calculs comme un calcul standard, le nombre de calculs à réaliser peut devenir très important, plusieurs milliers de calculs et leur mise en œuvre devient très fastidieuse.

L'objectif de ce document est de décrire la méthodologie à mettre en œuvre dans *Code_Aster* pour réaliser ce type d'étude avec le minimum d'intervention.

Globalement, une étude paramétrique est la donnée d'une étude nominale (fichier de commande commun à chaque paramètre) et d'un jeu de paramètres : *Code_Aster* va décliner l'étude nominale en plusieurs études en fonction des paramètres fournis, puis va exécuter chaque déclinaison en prenant en compte les ressources machine disponibles.

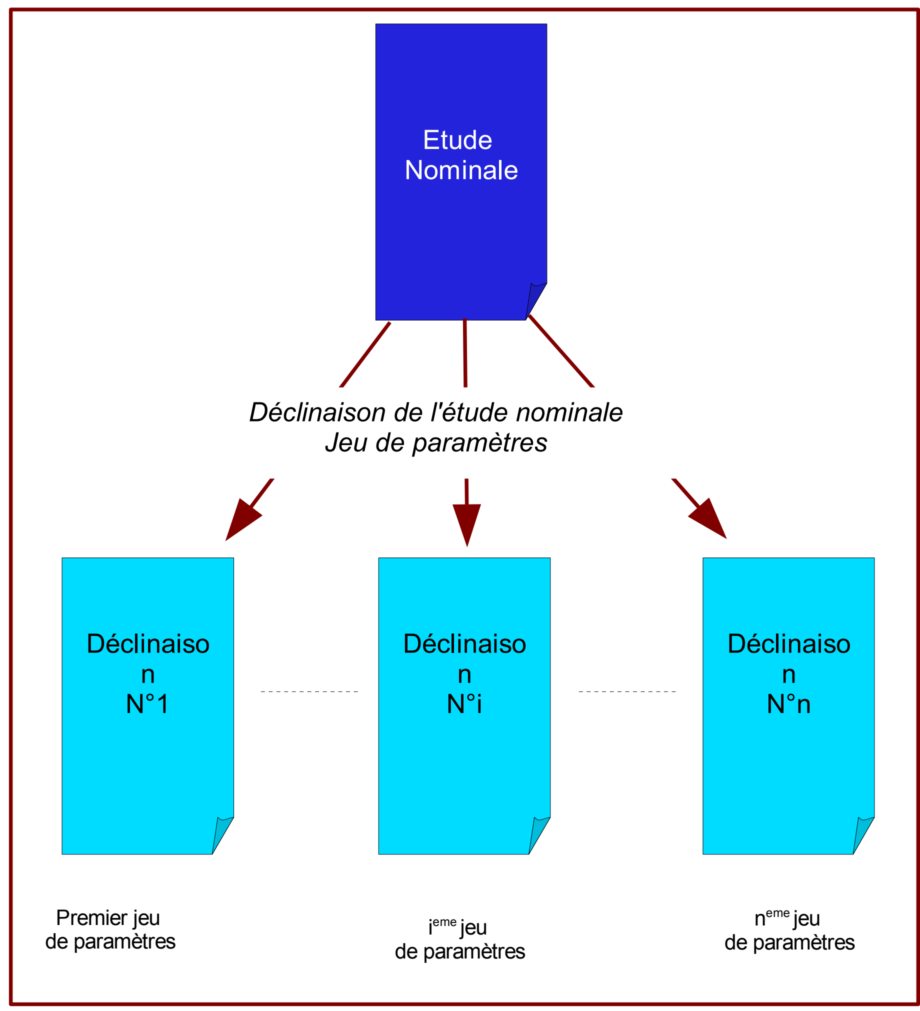


Figure 1.1-a: Déclinaisons

1.2 Objet du tutoriel

L'objectif du tutoriel est de fournir à l'utilisateur un maximum d'informations pour lui permettre de mettre en œuvre des études paramétriques.

Ce document répond aux questions suivantes :

- Comment préparer une étude paramétrique (fichiers,...),
- Comment lancer une étude paramétrique (répertoires, options,...),
- Comment générer et relire une base,
- Comment utiliser les ressources calculs disponibles (distribution des calculs,...),
- Comment récupérer les résultats de l'étude paramétrique (fichier de commandes, répertoire de résultats).

Dans le dernier paragraphe on présente un exemple détaillé d'une étude paramétrique.

Dans ce document, on suppose que l'utilisateur est familiarisé avec les calculs standard *Code_Aster*. Dans le cas contraire il est invité à consulter les documents suivants [U1.04.00], ... et à commencer par effectuer une étude classique .

2 Mode de fonctionnement des calculs paramétriques

2.1 Principe

La mise en œuvre d'études paramétriques dans *Code_Aster* est relativement simple. Néanmoins il est important que :

- votre étude standard tourne sans problème avant d'entreprendre ce type de calcul.
- vous ayez bien défini au préalable les paramètres que vous souhaitez faire varier:
  - leurs noms,
  - leurs valeurs,
  - les scénarios de calculs à exécuter.

Dans le cas d'une étude standard vous disposez au minimum :

- En donnée :
  - d'un fichier de commande (`.comm`)
  - des fichiers de données nécessaires (`.mail`, `.med`,...)
  - des paramètres (temps et mémoire)
- En sortie :
  - d'un fichier message (`.mess`)
  - d'un fichier résultat (`.resu`, `.med`,...)

Pour effectuer une étude paramétrique vous avez besoin en plus :

- En donnée :
  - d'un fichier décrivant le jeu de paramètre (`.distr`)
  - d'un répertoire résultats (`.repe`)
  - d'une option Astk `distrib=oui`. Selon le serveur, on peut avoir besoin de spécifier la classe batch dans laquelle les calculs seront soumis.
  - les paramètres (temps/mémoire) sont les mêmes pour tous les calculs et égaux à ceux du job maître.
- En sortie :
  - d'un répertoire résultats (`.repe`)

Dans les paragraphes suivants nous reprenons tous ces points en détails.

Dans le tableau suivant nous avons résumé les éléments obligatoires et facultatifs que l'on retrouve dans le cas d'une étude paramétrique.

Éléments	Études standards	Études paramétriques	Commentaires	§
Fichier de commande	c	c	- Définit le jeu de commande - Utilisation des paramètres	
Fichier de maillage	c	c	Définition du maillage	
distr	x	p	Définition des paramètres et des scénarios de calcul : valeur des paramètres, soit de façon explicite soit de façon calculé	2.2.1
.mess	c	x	Fichier message	
.resu	c	x	Fichier résultat au format aster	
.rmed,...	c	p	Fichier résultat au format med	
repe_out	c	p	Répertoire des résultats	
hostfile	x	p	Permet de lancer des calculs parallèles	2.4
astk	c	p	Options de lancement **distrib = oui**	2.3
base	c	p	Les bases sont stockées dans le repe_out	2.4
poursuite	c	c		2.4

x : pas de fichier
c : fichier classique
p : fichier paramétrique

Tableau 2.1-1: Synthèse des éléments obligatoires et facultatifs dans une étude paramétrique

2.2 Mise en œuvre

Les actions à entreprendre pour mettre en œuvre un calcul paramétrique sont les suivantes:

1) Écriture d'un jeu de paramètres,
2) Utilisation des paramètres dans le fichier de commande,
3) Sous Astk onglet `ETUDE` : définition du répertoire résultats et ajout du jeu de paramètres,
4) Sous Astk menu `OUTILS` : définition du type de calcul

2.2.1 Écriture d'un jeu de paramètres

La description, en langage python, du jeu des paramètres est effectuée dans le fichier `'.distr'`. dans lequel on trouve

- la liste des paramètres,
- les valeurs que vont prendre successivement ces paramètres (scénarios de calculs).

Exemple :

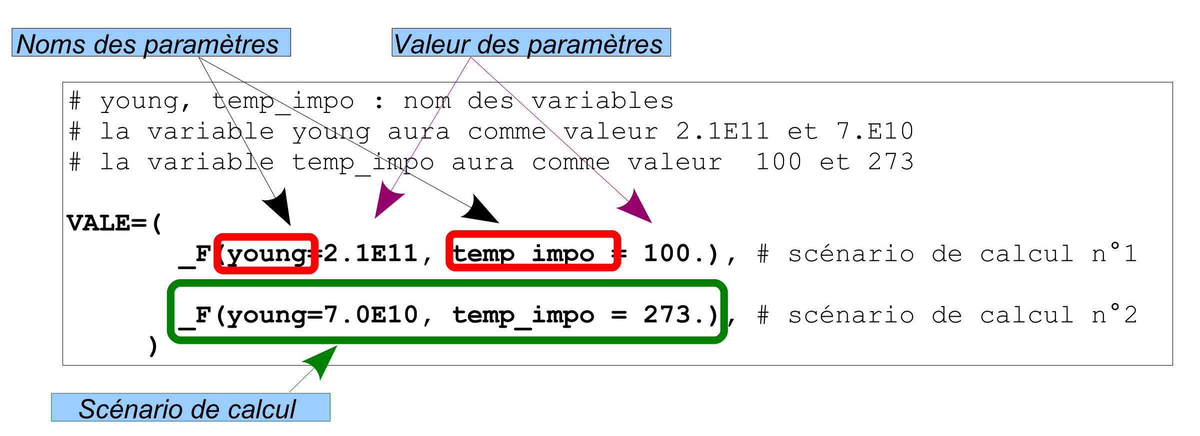


Figure 2.2.1-a: Exemple de Fichier "distr" explicite

Dans cet exemple, on a deux scénarios de calcul (n°1 et n°2) avec les paramètres `young` et `temp_impo` qui prendrons successivement comme valeur $(2.1\text{E}11, 100)$ et $(7.0\text{E}10, 273)$.
A ce stade, bien que le nom des paramètres soit suffisamment explicites, rien ne permet de dire comment ils seront utilisés. Pour le paramètre `young`, cela laisse supposer que celui-ci sera utilisé lors de la définition du matériau, mais rien pour l'instant ne permet de l'affirmer.

Du point de vue langage python, le fichier `'.distr'` contient une liste de dictionnaires de nom `VALE`, (chaque dictionnaire correspondant à un calcul). On utilise `_F` comme dans le fichier de commande pour définir ces dictionnaires.

La définition des paramètres des scénarios de calcul peut être donnée de façon explicite ou calculée.

Remarque

Dans l'exemple précédent, la définition des paramètres et les scénarios de calculs sont donnés explicitement. Il est possible de faire appel aux fonctionnalités du langage python pour les définir de façon automatique, un exemple pratique est présenté dans le §3.2.2).Le nom des paramètres est défini par l'utilisateur, il est conseillé d'utiliser des noms explicites.

2.2.2 Utilisation des paramètres dans le fichier de commande

Dans le paragraphe §2.2.1, on a défini le jeu de paramètres (nom, valeurs et scénarios de calcul), l'objectif maintenant est d'utiliser ces paramètres dans le fichier de commande, à l'endroit souhaité.

La prise en compte de ces paramètres dans le fichier de commande s'effectue en deux temps :

1) déclaration des variables python : les paramètres
2) utilisation de ces variables dans le fichier de commandes.

- **Déclaration des paramètres (variables python)**

Avant toute utilisation dans une commande ou dans une expression quelconque, il est **obligatoire** que les paramètres soient connus. L'objectif n'est pas de les initialiser une nouvelle fois, mais simplement de les déclarer en tant que variables python.

Exemple : dans le fichier `.distr` précédent, on a défini les paramètres `young` et `temp_impo`. On trouvera donc dans le fichier de commande les lignes suivantes:

```
DEBUT()
...
young = 0.
temp_impo = 0.
...
FIN()
```

Figure 2.2.2-a: Déclaration des paramètres

Remarques:

Les valeurs affectées dans le fichier de commande ne seront pas utilisées, elles seront remplacées automatiquement par celles définies dans le fichier `'distr'`*, pour chacun des scénarios de calculs.*
Évitez de choisir un nom de paramètre identique à un mot-clé de Code_Aster. La substitution étant faite par l'expression régulière `'^( *)nom *= *'`*, il pourrait y avoir confusion. La casse étant signifiante, utiliser des noms en minuscules permet d'éviter cet écueil.*

- **Utilisation des paramètres**

L'utilisation des paramètres dans le fichier de commande s'effectue de manière classique, au sein des commandes, des expressions mathématiques, ...

Exemple :

```
DEBUT()
...
young=0.
temp_impo=0.
...
mat=DEFI_MATERIAU(ELAS=_F(E=young,...))
...
t0 = AFFE_CHAR_MECA=(TEMP_IMPO=
                          _F(TEMP=temp_impo,
                             GROUP_NO='CHAUD',..)
...

FIN()
```

Figure 2.2.2-б: Fichier de commande

2.2.3 Type des paramètres

Le type des paramètres peut être quelconque. Ensuite, il faut garder à l'esprit que le remplacement est textuel lors de l'instanciation du jeu de commandes pour un jeu donné de paramètres.

Exemple :

Soit le jeu de paramètres :

```
VALE=(_F( young  =   2.1e11,
          indice =   4,
          nomp   =   "'INST'",
          objf   =   'FON1',),
...)
```

et l'entête du jeu de commandes :

```
young  =   2.e11
indice =   0
nomp   =   'X'
objf   =   FON0
...
```

Le jeu de commandes décliné sur le jeu de paramètres sera :

```
young  =   2.1e11
indice =   4
nomp   =   'INST'
objf   =   FON1
...
```

La différence est visible pour les deux derniers paramètres où l'on voit qu'on a utilisé le texte de la chaine de caractères de `nomp` et `objf` (et non la représentation de celle-ci, donc un jeu de cotes/guillemets a disparu).

Cela permet de paramétriser l'utilisation de concepts. Ici, on utilise la fonction `FON1` comme paramètre `objf`.

2.3 Lancement des études : Astk

Le lancement des études paramétriques est géré directement par Astk. Ce lancement est identique à celui d'une étude classique moyennant l'ajout d'informations complémentaires telles que :

- ajout dans le profil d'étude d'une ligne pour prendre en compte le fichier '`distr`'
- ajout dans le profil d'étude d'une ligne pour définir le répertoire résultats de type `'repe'`
- initialisation dans le menu option, du paramètre `distrib` à `'OUI'`.

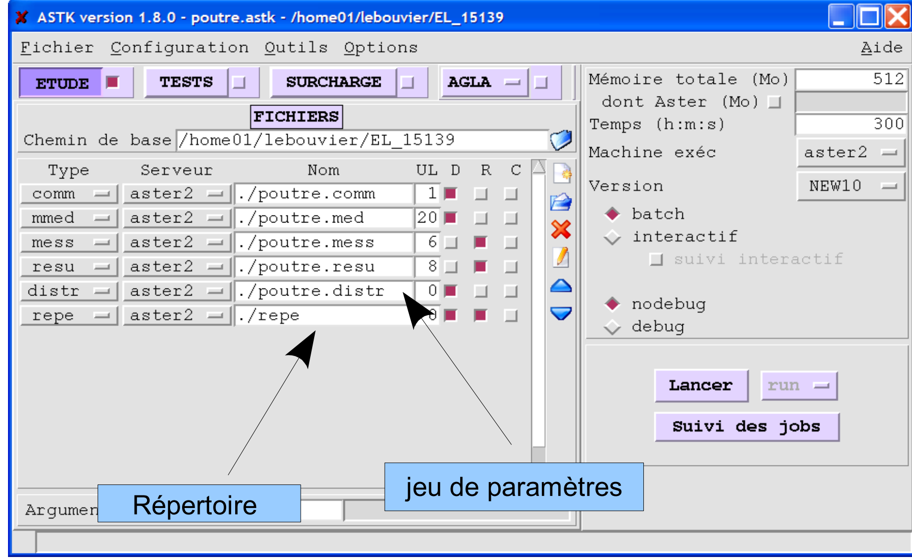


Figure 2.3-a: Définition du jeu de paramètres, répertoire (astk)

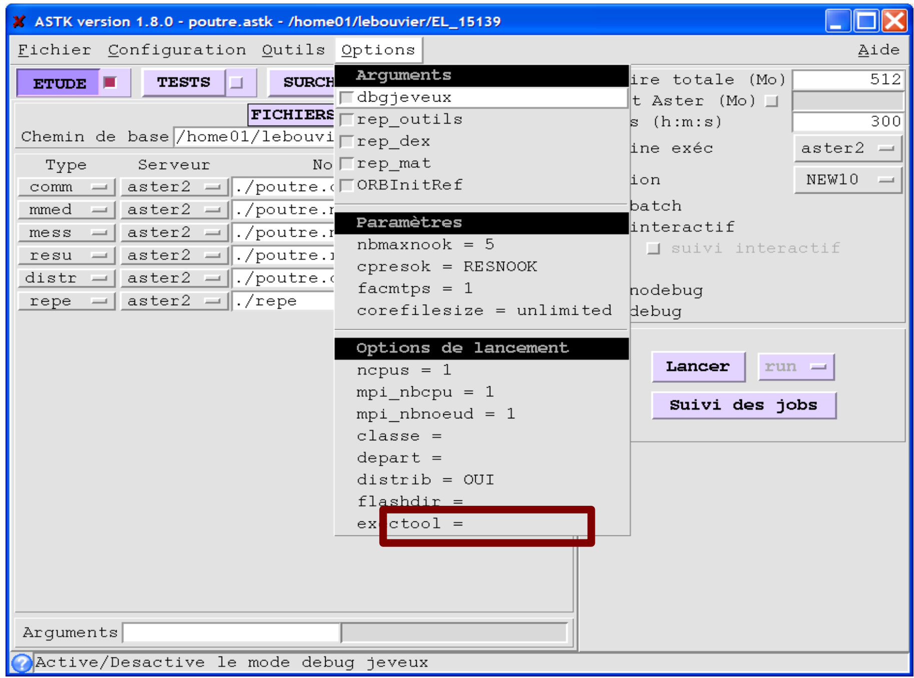


Figure 2.3-b: option distrib (`Astk`)

Remarques

Il est possible, comme tous les calculs Code_Aster, de lancer les études paramétriques en mode ligne de commande avec `as_run`. Pour cela, il est nécessaire de disposer au préalable du fichier `.export` :

```
as_run --serv nom_etude.export
```

2.3.1 Gestion des calculs et des résultats

Le répertoire de calculs est commun à celui des résultats : il est défini dans le profil d'étude sous le type `'repe'`. Après l'exécution des calculs paramétriques, on trouve dans ce répertoire autant de répertoires `calc_i` que de calculs lancés. On trouve également un répertoire `flash`. Le détail de chacun de ces répertoires est présenté ci-dessous.

2.3.1.1 Gestion des calculs

Les fichiers de commande correspondant à chacun des scénarios sont déclinés automatiquement à partir du fichier de commande nominal et du jeu de paramètres. Ces fichiers sont stockés dans les répertoires `calc_i`.
Les fichiers output (message et erreur) de chaque calcul sont stockés dans un seul répertoire, le répertoire `flash`. Toutes les informations concernant le déroulement de chaque exécution se trouvent dans ce répertoire.

2.3.1.2 Gestion des résultats

Comme pour un calcul classique, l'utilisateur a la possibilité de générer des fichiers de sortie (tables, graphes, ...). Il devra les définir dans le fichier de commande standard. Ces sorties seront communes à tous les scénarios de calcul.
Par exemple, supposons que l'utilisateur veuille imprimer une table dans le fichier d'unité logique 38 et un résultat au format med dans le fichier d'unité logique 39. Il devra définir dans le fichier de commande nominal, le nom de chacun des fichiers de sortie afin qu'ils puissent être stockés dans le répertoire `REPE_OUT` produit automatiquement par *Code_Aster.* L'extrait du fichier de commande ci-dessous illustre cette définition :

```
DEFI_FICHIER(UNITE=38,FICHIER='./REPE_OUT/table.out')
IMPR_TABLE(TABLE=SIYY,UNITE=38,NOM_PARA=('NOEUD','SIYY'));

DEFI_FICHIER(UNITE=39,FICHIER='./REPE_OUT/poutre.rmed')
IMPR_RESU(FORMAT='MED',UNITE=39,RESU=_F(RESULTAT=depl))
```

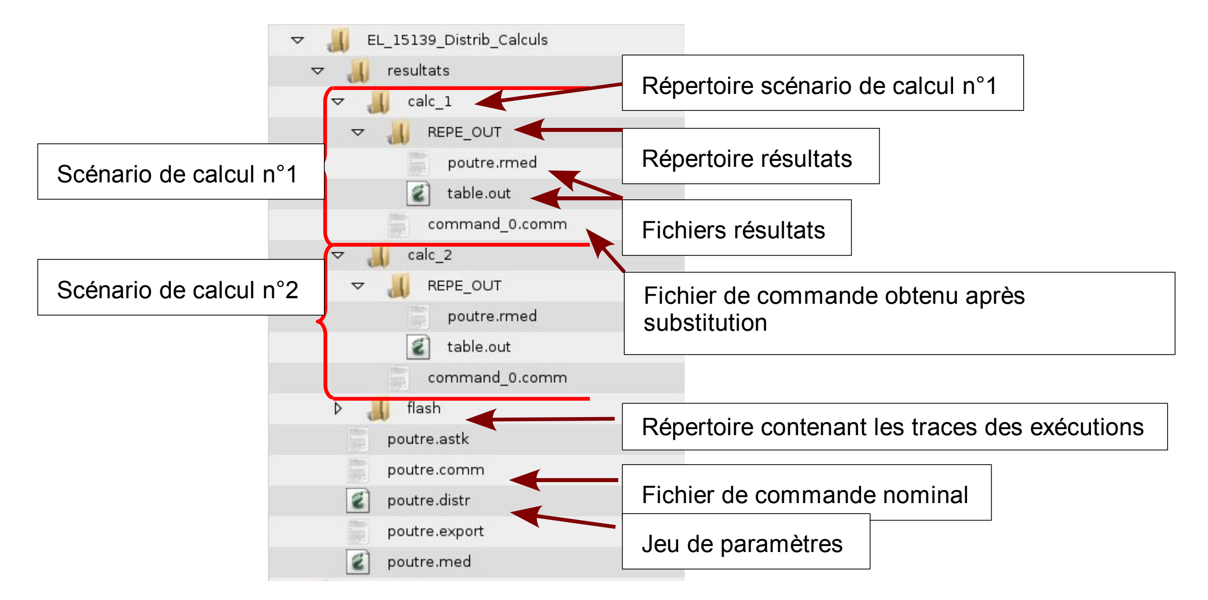
Figure 2.3.1.2-a: définition des fichiers de sortie et de leurs unités logiques

Remarques:

Dans le cas d'étude paramétrique les fichiers `.mess` et `.resu` ne sont pas créés. Toutefois, la consultation des fichiers du répertoire `flash` permet de recueillir les informations émises d'ordinaire dans le `.mess`.

2.3.1.3 Arborescence des répertoires

La figure ci-dessous présente la localisation des fichiers et répertoires de 2 scénarios de calcul.



2.4 Fonctionnalités complémentaires

2.4.1 Générer une base

Comme pour une étude standard, il est possible de générer une base. Pour cela, il suffit d'ajouter dans le profil d'étude une entrée de type `'base'` avec pour nom celui du répertoire `repe`. On trouvera pour chacun des scénarios de calcul la base correspondante stockée dans le répertoire `calc_i/base`.

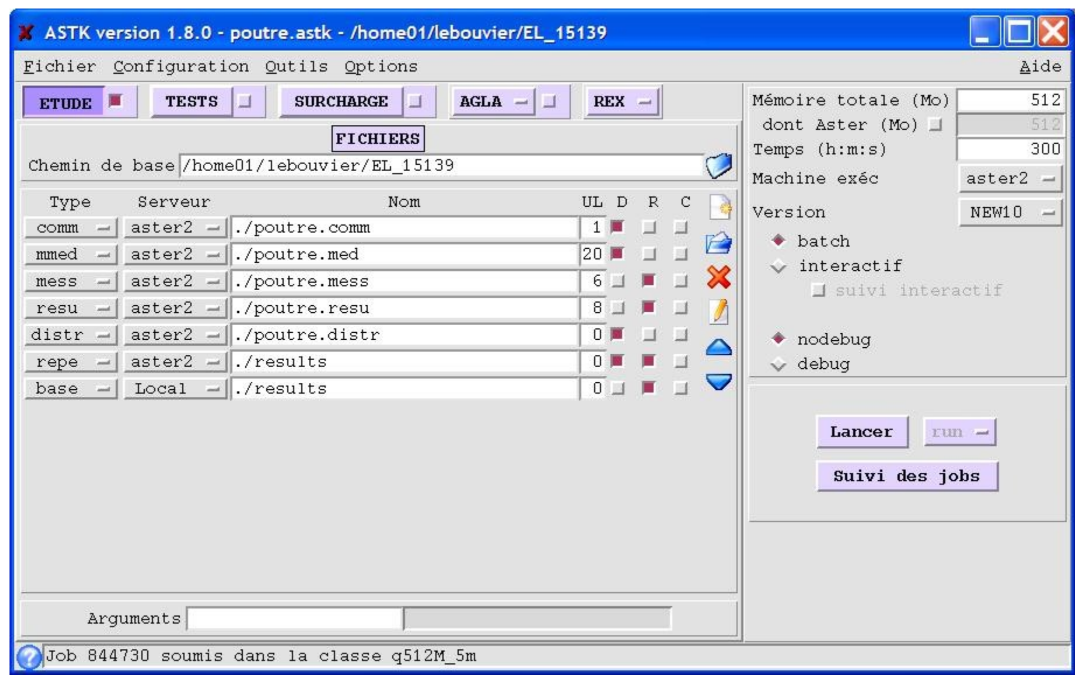


Figure 2.4.1-a: Exemple de profil d'étude pour générer une base par calcul

2.4.2 Effectuer une poursuite

Comme pour une étude standard, il est possible d'exploiter une base. Pour cela, il suffit d'ajouter dans le profil d'étude une entrée de type `'base'` avec pour nom celui du répertoire `repe` dans lequel chaque base à lire est présente dans le répertoire `calc_i/base`.

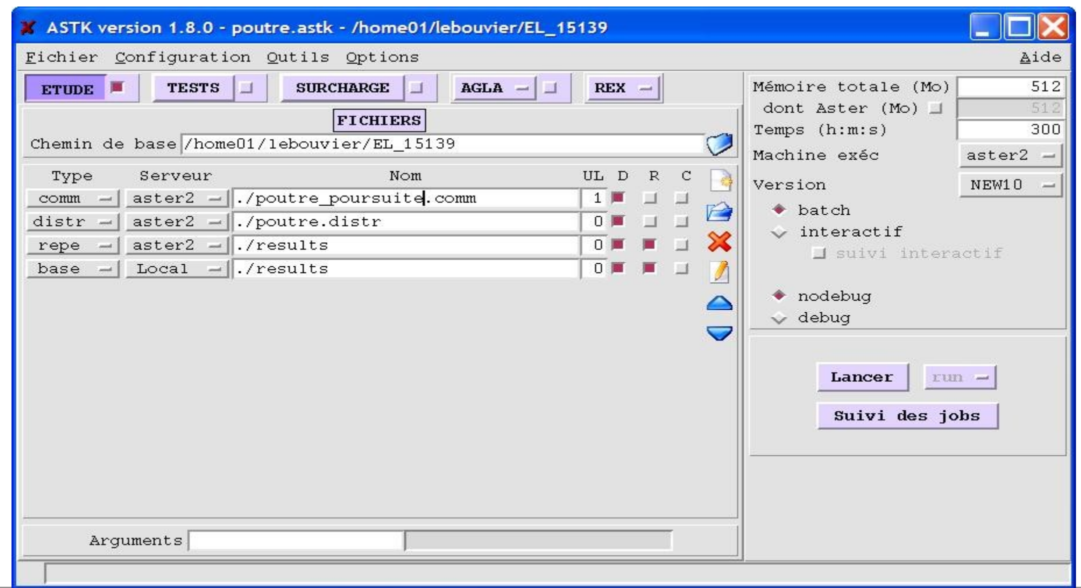
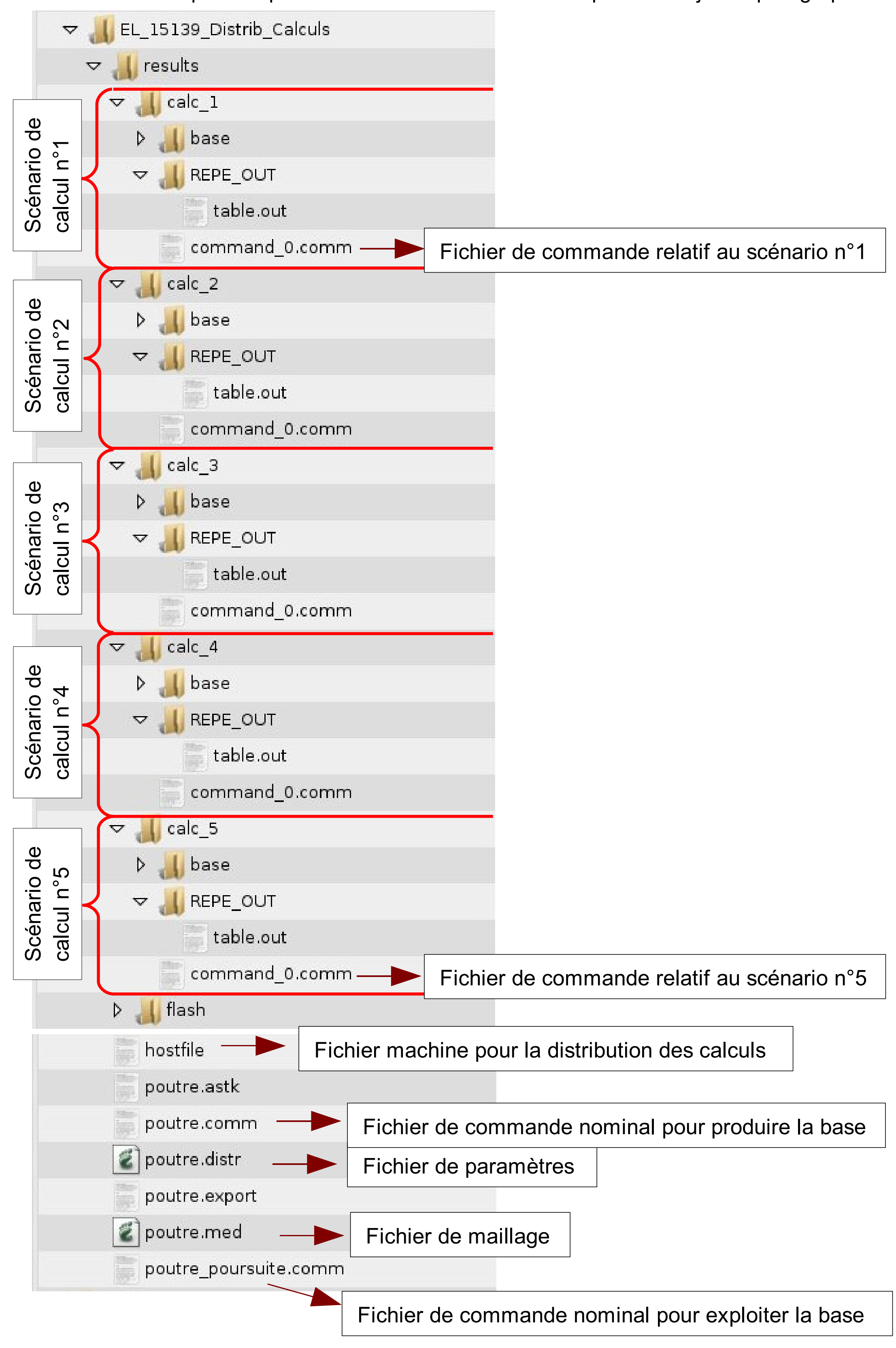


Figure 2.4.2-a: Exemple de profil d'étude pour exploiter la base associée à chaque calcul

2.4.3 Arborescence des répertoires

On présente l'arborescence des répertoires en présence d'une lecture de base pour chacun des 5 calculs. On remarquera la présence d'un fichier `hostfile` qui fera l'objet du paragraphe suivant.



2.4.4 Distribution de calculs

Dans le cas d'études paramétriques, les calculs sont indépendants les uns des autres. Il est donc possible d'utiliser les ressources machine disponibles lors de la distribution des calculs. Il convient donc de créer un fichier de type `'hostfile'` définissant les ressources machine. On y définit:

- le nom du nœud où seront soumis les jobs,
- le nombre maximum de job à soumettre en même temps,
- la mémoire allouée totale.

Remarque:

> *Sur un serveur partagé, il est conseillé d'utiliser le fichier défini par l'administrateur et donc, de ne pas redéfinir son propre fichier de ressources.*
> *Si le fichier `hostfile` n'est pas présent dans le profil d'étude, c'est le fichier `batch_distrib_hostfile` présent dans le répertoire `etc/codeaster` qui sera utilisé par défaut.*

Par exemple sur le serveur centralisé, les ressources sont gérées par le logiciel de batch. Dans ce cas, le fichier `hostfile` déclare simplement le nombre de calculs qui seront soumis en même temps. Ici on peut soumettre jusqu'à 32 calculs sur chaque nœud frontal (la mémoire est indiquée *infinie*, c'est à dire qu'on laisse le logiciel de batch gérer) :

```
[ataster1]
cpu=32
mem=9999999

[ataster2]
cpu=32
mem=9999999
```

Figure 2.4.4-a: Exemple de fichier `hostfile` avec serveur de batch

Pour que votre fichier de ressource soit pris en compte, il suffit de l'ajouter dans votre profil d'étude en précisant le type `'hostfile'`.

Dans le cas d'utilisation d'un ensemble de machines disponibles en interactif, le fichier `hostfile` pourrait ressembler à :

```
# nom du nœud
[machine1]
# nombre de CPU disponible
cpu=4
# mémoire totale de la machine en Mo
mem=4000

# nom du nœud
[machine2]
# nombre de CPU disponible
cpu=8
# mémoire totale de la machine en Mo
mem=4000
```

Figure 2.4.4-b: Exemple de fichier `hostfile` en interactif

On pourra avoir jusqu'à 12 calculs exécutés simultanément selon la mémoire disponible répartis sur les deux machines. Si chaque calcul requiert 2 Go de mémoire, il y aura au maximum deux calculs par machine pour ne pas dépasser le montant total de mémoire disponible.

2.4.5 Pré/post-traitements communs à tous les calculs

On dispose de 4 mots-clés dans le fichier « distr » : PRE_CALCUL, UNITE_PRE_CALCUL, POST_CALCUL, UNITE_POST_CALCUL.
PRE_CALCUL (resp. POST_CALCUL) définit un texte (un ensemble de commandes Aster) qui sera inclus juste après DEBUT (resp. juste avant FIN).
UNITE_PRE_CALCUL (resp. UNITE_POST_CALCUL) propose le même fonctionnement sauf qu'on fournit un numéro d'unité logique.

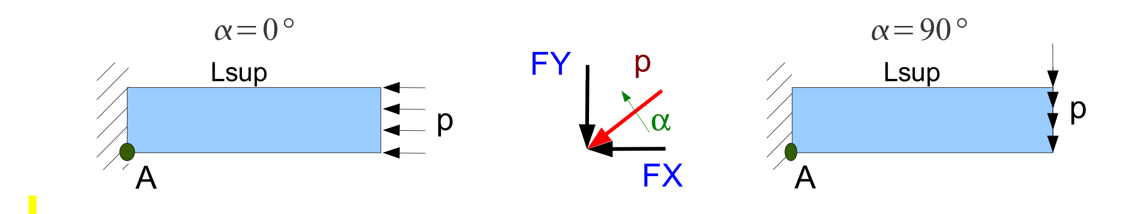
Cette modification est faite pour tous les fichiers de type « comm » présents dans le profil.

Cette possibilité est notamment utilisée par la commande de recalage MACR_RECAL pour ajouter un post-traitement à chaque calcul esclave.

3 Exemples d'application

3.1 Présentation

L'exemple retenu pour présenter la mise en œuvre d'études paramétriques est présenté sur la figure ci-dessous. Il s'agit d'une poutre encastrée à une extrémité et soumise à une pression répartie à l'autre extrémité. L'objectif de cette étude paramétrique est de déterminer l'évolution de la contrainte équivalente de Von Mises le long de la ligne supérieure de la poutre en fonction de l'orientation α .
Les fichiers de données sont ceux du cas-test `distr01a`.



Les calculs seront effectués tous les $22.5°$, pour un angle α variant de $0°$ à $90°$. Le nombre de scénarios de calcul à réaliser est donc de 5. Dans le tableau ci-dessous nous présentons les valeurs que vont successivement les paramètres FX et FY .

Scénario n°	Angle α	$FX=P.cos(a)$	$FY=P.sin(a)$
1	0.0°	10^6	$0.$
2	22.5°	9.23879×10^5	-3.82683×10^5
3	45.0°	7.07106×10^5	-7.07106×10^5
4	67.5°	3.82683×10^5	-9.23879×10^5
5	90.0°	$0.$	-10^6

Tableau 3.1-1: Valeurs des paramètres

Le fichier de commande de l'étude nominale est `distr01a.comm`.
Dans la suite de cette présentation, nous allons détailler la mise en œuvre de ce type de calcul, à savoir :

- la génération du fichier `'distr'`
- l'utilisation des paramètres dans le fichier de commande.

3.2 Définition du jeu de paramètres et des cas de calculs

Il est possible de définir les paramètres dans le fichier `'.distr'` de deux façons :

- Explicitement : dans ce cas l'utilisateur fournit toutes les valeurs que peut prendre les paramètres.
- Calculés: dans ce cas l'utilisateur a recours à de la programmation python
  - pour calculer automatiquement ces paramètres sous forme conditionnelle ou pas,
  - pour définir automatiquement ces scénarios de calcul, par exemple en balayant toutes les combinaisons de paramètres possible ou en choisissant les valeurs min/moy/max des paramètres.

3.2.1 Fichier 'distr' explicite

Dans ce cas, on écrit explicitement les valeurs des paramètres dans le fichier 'distr'. Dans le cas de l'exemple, il se présente sous la forme suivante (fichier distr01a.50 du cas-test) :

```
VALE=(
       _F(F_Norm=1.E6,      F_Tang=0.),          # cas de calcul n°1
       _F(F_Norm=9.23879E5,F_Tang=-3.82683E5), # cas de calcul n°2
       _F(F_Norm=7.07106E5,F_Tang=-7.07106E5), # cas de calcul n°3
       _F(F_Norm=3.82683E5,F_Tang=-9.23879E5), # cas de calcul n°4
       _F(F_Norm=0.,        F_Tang=-1.E6),      # cas de calcul n°5
     )
```

Figure 3.2.1-a: Fichier 'distr' explicite

3.2.2 Fichier 'distr' calculé

Dans ce cas, l'écriture du fichier 'distr' est moins simple, on fait appel à la programmation du langage python. Pour cet exemple, il se présente sous la forme suivante (fichier distr01a.51 du cas-test) :

```
from math import pi, cos, sin
import numpy

VALE = []
n = 5
list_theta = numpy.arange(n) * 22.5 * pi / 180.
P = 1.e6

for a in list_theta:
   VALE.append(_F(F_Norm = P*cos(a),
                  F_Tang = P*sin(a),))
```

Figure 3.2.2-a: Fichier 'distr' calculé

3.3 Utilisation des paramètres dans le fichier de commande

Il suffit de référencer, dans le fichier de commande nominal, les noms des paramètres présents dans le fichier 'distr'.

```
DEBUT()

# Initialisation (ici apparaitront les valeurs des paramètres)
F_Norm=0.
F_Tang=0.

...

CHAR=AFFE_CHAR_MECA(MODELE=MODE,
                    FORCE_CONTOUR=_F(GROUP_MA = 'Press',
                                     FX = F_Norm ,
                                     FY = F_Tang),)
```

3.4 Post-traitements

Le fichier `distr01a.11` donne un exemple de post-traitement avec relecture de l'ensemble des fichiers résultats, fusion des résultats dans une table unique, impression d'une courbe avec l'ensemble des évolutions de la contrainte ...

4 Conseils d'utilisation

Voici quelques conseils :

- L'étude standard doit est valide avant d'être déclinée sur le jeu des paramètres. Elle doit tourner sans erreur.
- Avant d'être déclinée sur le jeu des paramètres, il est aussi important d'optimiser les calculs en diminuant le temps d'exécution, ...
- Vérifier que le fichier de paramètres `'distr'` est correct.
- Attention à l'espace disque : si le nombre de calculs et leur taille sont importants, la sauvegarde des bases peut prendre beaucoup de place.
- Vérifier la définition des variables dans le fichier de commande, est-elle correcte ?
- Tester le bon fonctionnement de votre étude paramétrique en utilisant un fichier `'distr'` qui ne définit qu'un seul jeu de paramètre.

5 Questions/Réponses

Questions	Réponses
Où définit-on les paramètres?	Dans le fichier : • `.distr` : on y définit leurs noms et leurs valeurs • `.comm` : on y définit le nom variable python
Où définit-on les scénarios de calculs?	Dans le fichier `'distr': VALE(_F(...)` `_F(...))`
Que doit-on faire sous Astk?	Au minimum : • définir une ligne pour le fichier `'distr'` • définir une ligne pour le répertoire `'repe'` • définir l'option de lancement `distrib=oui`
Ou trouve t' on les résultats?	Dans le répertoire `repe`

Je ne trouve pas mes résultats !	Avez-vous dans le fichier de commande créé la commande suivante:
	`DEFI_FICHIER(`
	`UNITE=numero_unite_logique,`
	`FICHIER='.REPE_OUT/nom_fichier')`

Je ne trouve pas de fichier `.mess` et `.resu`	Ces fichiers n'existent pas, par contre vous pouvez consulter dans le répertoire `repe/flash` les fichiers décrivant l'exécution de chacun des scénarios.
Mes calculs ne fonctionnent pas ?	Analyser les messages éventuellement émis à l'écran, consulter les fichiers output et erreur présent dans le répertoire flash